



## CSCE 421: Machine Learning

### Lecture 3

## Outline

- Perceptron
- Example & Representation
- Perceptron Learning Algorithm (PLA)

(\* Part of the following slides are taken from Dr. Malik Magdon-Ismail's Machine Learning class.)

## Perceptron: Example

### Credit approval or denial

- Task: Approve or deny credit (binary classification task)
- Features: Salary, debt, years in residence, etc.



## Perceptron: Example & Representation

- Input vector  $\mathbf{x} = [x_1, \dots, x_d]^T$ .
- Give importance weights to the different inputs and compute a “Credit Score”

$$\text{“Credit Score”} = \sum_{i=1}^d w_i x_i.$$

- Approve credit if the “Credit Score” is acceptable.

$$\text{Approve credit if } \sum_{i=1}^d w_i x_i > \text{threshold}, \quad (\text{“Credit Score” is good})$$

$$\text{Deny credit if } \sum_{i=1}^d w_i x_i < \text{threshold}. \quad (\text{“Credit Score” is bad})$$

- How to choose the importance weights  $w_i$ 
  - input  $x_i$  is important  $\implies$  large weight  $|w_i|$
  - input  $x_i$  beneficial for credit  $\implies$  positive weight  $w_i > 0$
  - input  $x_i$  detrimental for credit  $\implies$  negative weight  $w_i < 0$

## Perceptron: Example & Representation

$$\begin{aligned} \text{Approve credit if } \sum_{i=1}^d w_i x_i &> \text{threshold,} \\ \text{Deny credit if } \sum_{i=1}^d w_i x_i &< \text{threshold.} \end{aligned}$$

can be written formally as

$$h(\mathbf{x}) = \text{sign} \left( \left( \sum_{i=1}^d w_i x_i \right) + w_0 \right)$$

The “bias weight”  $w_0$  corresponds to the threshold. (How?)

## Perceptron: Representation

We have defined a *Hypothesis set*  $\mathcal{H}$

$$\mathcal{H} = \{h(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x})\} \quad \leftarrow \text{uncountably infinite } \mathcal{H}$$

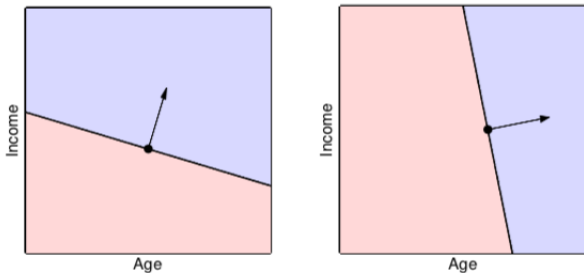
$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \in \mathbb{R}^{d+1}, \quad \mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \in \mathbb{R}^{d+1}$$

This hypothesis set is called the *perceptron* or *linear separator*

## Perceptron: Representation

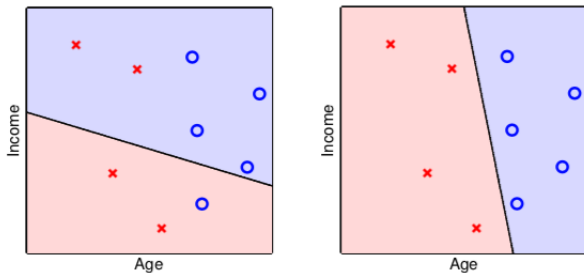
$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x})$$

(Problem 1.2 in LFD)



How to find the perpendicular vector to a line?  
 See Handout 4 on Piazza

## Perceptron: Representation



A perceptron fits the data by using a line to separate the  $+1$  from  $-1$  data.

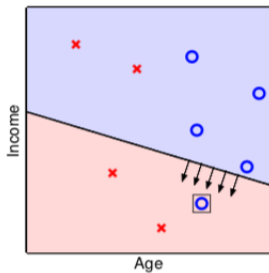
**Fitting the data:** How to find a hyperplane that *separates* the data?

(“It’s obvious - just look at the data and draw the line,” is not a valid solution.)



## Perceptron Learning Algorithm

**Idea!** Start with some weight vector and try to improve it.



## Perceptron Learning Algorithm: Example

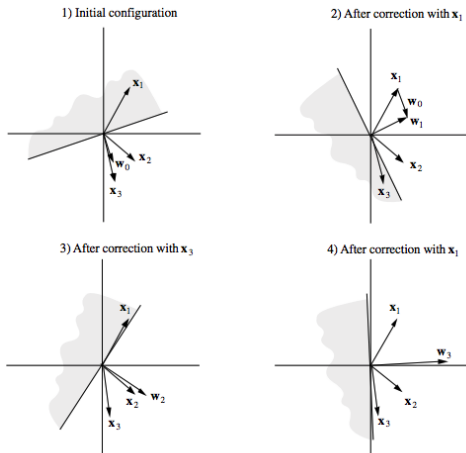


Fig. 4.10. Convergence behavior of the learning algorithm

Assuming that  $\{x_1, x_2, x_3\}$  belong to the same class, after the initial updates, successive corrections become smaller and the algorithm “fine tunes” the position of the weight vector.

## Perceptron Learning Algorithm

### A simple iterative method

#### Incremental learning on single example at a time

- 1 Initialization  $\mathbf{w}(\mathbf{0}) = \mathbf{0}$  (or any other vector)
- 2 for  $t = 1, 2, 3, \dots$ 
  - a From  $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$  pick a misclassified sample
  - b Call the misclassified sample  $(\mathbf{x}_s, y_s)$ :  $\text{sign}(\mathbf{w}(\mathbf{t})^T \mathbf{x}_s) \neq y_s$   
 $(\mathbf{w}(\mathbf{t})^T \mathbf{x}_s = -1 \text{ if } y_s = 1; \mathbf{w}(\mathbf{t})^T \mathbf{x}_s = 1 \text{ if } y_s = -1)$
  - c Update the weight:  
 $\mathbf{w}(\mathbf{t} + \mathbf{1}) = \mathbf{w}(\mathbf{t}) + y_s \mathbf{x}_s$
  - d  $t \leftarrow t + 1$

## Perceptron Learning Algorithm

### Algorithm Learning Rate

- Vectors  $\mathbf{w}$  are not necessarily normalized
- If  $\|\mathbf{w}(t)\| \gg \|\mathbf{x}_s\|$  the new weight vector  $\mathbf{w}(t) + \mathbf{x}_s$  is almost equal to  $\mathbf{w}(t)$
- We can add a **learning rate**  $\alpha > 0$  in the update
  - $\mathbf{w}(t+1) = \mathbf{w}(t) \pm \alpha \mathbf{x}_s$
  - if  $\alpha \gg 0$ ,  $\mathbf{x}_s$  will have a large impact on the update
  - if  $\alpha \equiv 0$ ,  $\mathbf{x}_s$  will not influence much the update
  - Learning rate a typical **hyperparameter** of many learning algorithms and depends on the data of each problem

# Perceptron Learning Algorithm

## Algorithm Convergence

- The rule update considers a training sample at a time and may “destroy” the classification of other samples
- If the data can be fit by a **linear separator** (**linearly separable**), then after some finite number of steps, PLA is guaranteed to arrive to a correct solution.
- What if the data cannot be fit by a perceptron?
  - We can find infinitely many combinations of perceptrons that fit the data → neural networks

## Perceptron Learning Algorithm

### Algorithm Cost

- PLA is a local, greedy algorithm
- This can lead to an exponential number of updates of the weight
- In the following figure:
  - Two almost antiparallel vectors are to be classified in the same half-space
  - PLA rotates the separation line in one of the two directions and will require more and more time when the angle between the two vectors approaches 180 degrees

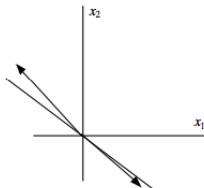


Fig. 4.13. Worst case for perceptron learning (input space)

## Parametric v.s. non-parametric models

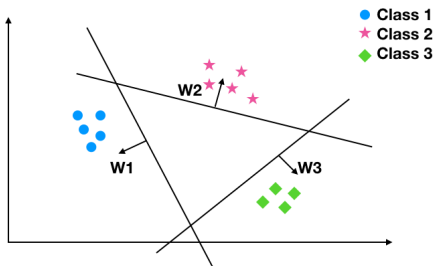
- Many possible ways to categorize learning models
- One way is to ask this question
  - Does the model have a fixed number of parameters or does it grow with the amount of training data?
- Non-parametric models (or instance/memory-based)
  - more flexible
  - computationally intractable for large datasets
  - e.g. K-NN
- Parametric models
  - faster to use
  - make strong assumptions about data
  - e.g. linear perceptron, linear regression

## Perceptron Learning Algorithm

### Question

Assume the following 2-dimensional data space with three classes  $C_1$ ,  $C_2$ ,  $C_3$ , and three linear classification boundaries  $\mathbf{w}_1$ ,  $\mathbf{w}_2$ , and  $\mathbf{w}_3$ . If  $\mathbf{x}_1 \in C_1$ ,  $\mathbf{x}_2 \in C_2$ , and  $\mathbf{x}_3 \in C_3$ , please select the equations that hold:

- (A)  $\mathbf{w}_1^T \mathbf{x}_1 > 0$ ,  $\mathbf{w}_2^T \mathbf{x}_2 > 0$ ,  $\mathbf{w}_3^T \mathbf{x}_3 > 0$
- (B)  $\mathbf{w}_1^T \mathbf{x}_1 < 0$ ,  $\mathbf{w}_2^T \mathbf{x}_2 < 0$ ,  $\mathbf{w}_3^T \mathbf{x}_3 < 0$
- (C)  $\mathbf{w}_1^T \mathbf{x}_1 > 0$ ,  $\mathbf{w}_2^T \mathbf{x}_2 < 0$ ,  $\mathbf{w}_3^T \mathbf{x}_3 < 0$
- (D)  $\mathbf{w}_1^T \mathbf{x}_1 < 0$ ,  $\mathbf{w}_2^T \mathbf{x}_2 > 0$ ,  $\mathbf{w}_3^T \mathbf{x}_3 > 0$



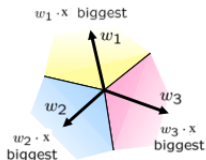
Correct answer is A.



## Multiclass perceptron

### Perceptron representation for more than two classes

- **Input:**  $\mathbf{x} \in \mathbb{R}^D$
- **Output:**  $y \in \{1, 2, \dots, K\}$
- **Training data:**  $\mathcal{D}^{train} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$
- **Model:**  $y = f(\mathbf{x}) = \arg \max_{k \in \{1, \dots, K\}} \mathbf{w}_k^T \mathbf{x}$
- **Model parameters:** weights  $\mathbf{w}_1, \dots, \mathbf{w}_K$



Learning occurs in the similar manner as the PLA for the 2-class perceptron.

## Three Components Of Learning

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
<i>K</i> -nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search
Hyperplanes	Likelihood	Branch-and-bound
Naive Bayes	Posterior probability	Continuous optimization
Logistic regression	Information gain	Unconstrained
Decision trees	K-L divergence	Gradient descent
Sets of rules	Cost/Utility	Conjugate gradient
Propositional rules	Margin	Quasi-Newton methods
Logic programs		Constrained
Neural networks		Linear programming
Graphical models		Quadratic programming
Bayesian networks		
Conditional random fields		

a learner must be  
represented in some  
formal language

an evaluation  
function assesses the  
performance of a  
learner

find the highest-  
scoring learner

## Summary

- Perceptron is a simple learning algorithm
  - decision boundary defined by a hyper-plane
  - hyper-plane is learned from the training data
- Reading materials
  - Abu-Mostafa 1.1.2